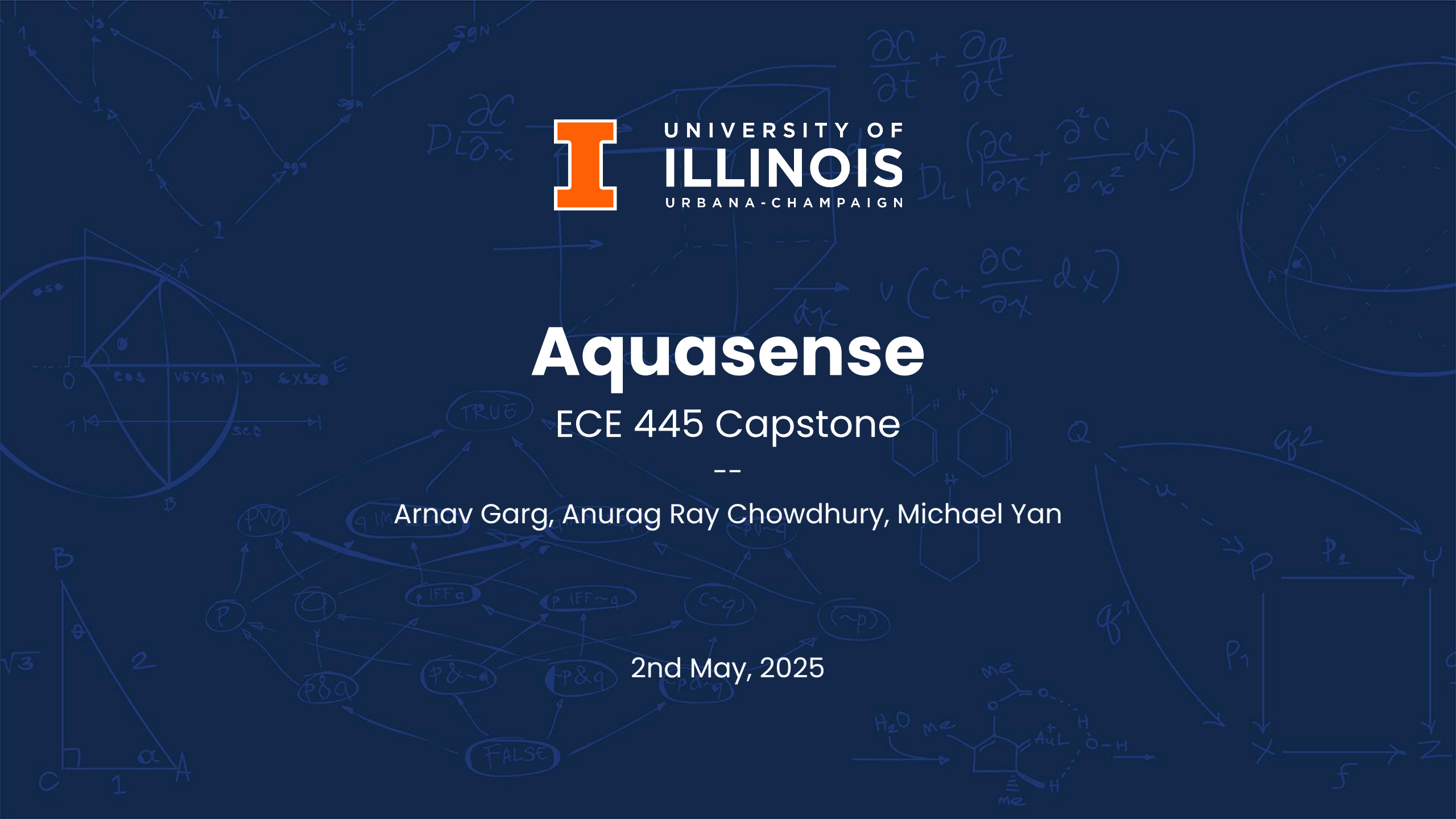


UNIVERSITY OF ILLINOIS URBANA-CHAMPAIGN

Aquasense

ECE 445 Capstone

--

Arnav Garg, Anurag Ray Chowdhury, Michael Yan

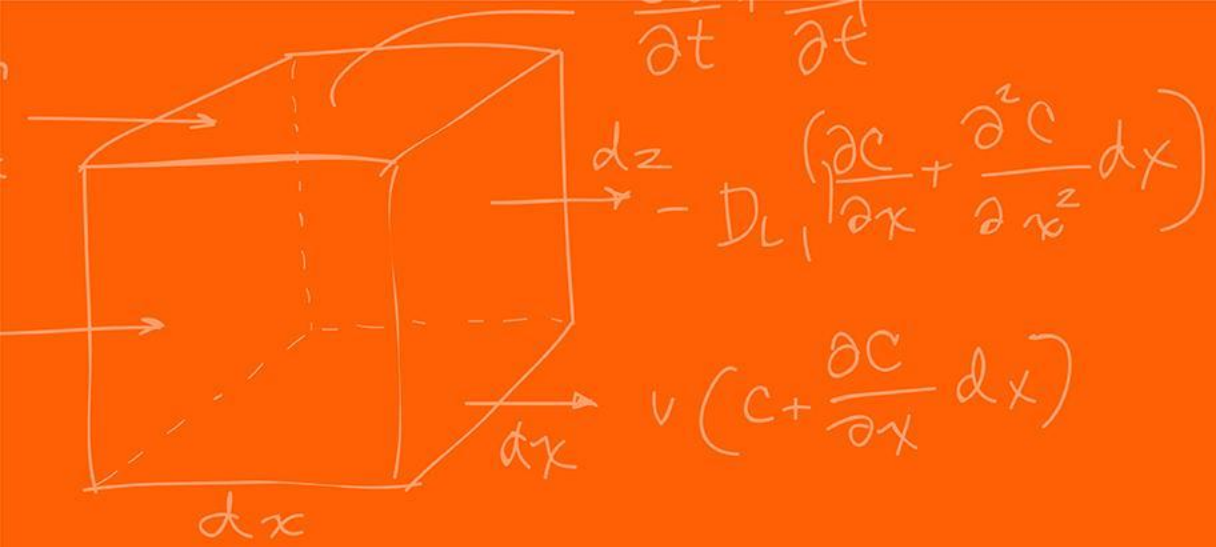
2nd May, 2025

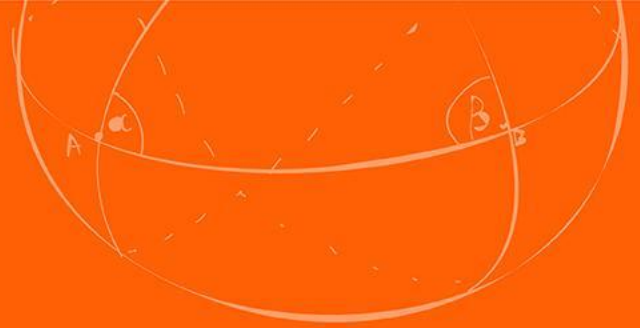
MEET THE TEAM

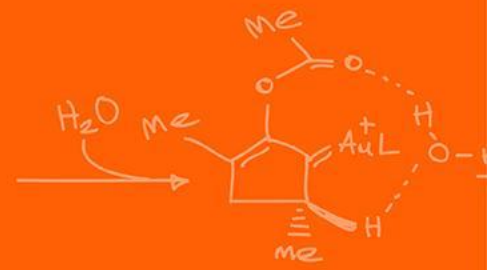
Arnav Garg

Anurag Ray C.

Michael Yan

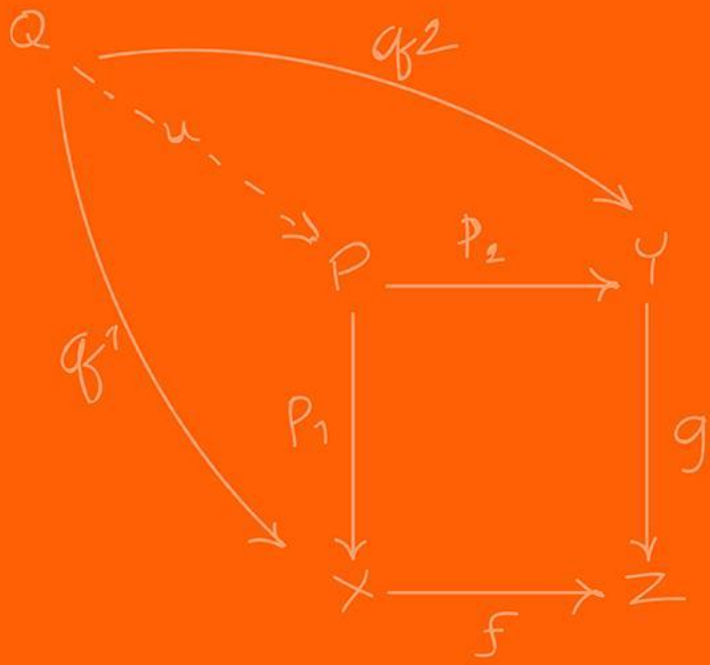






Purpose

Why Aquasense?



Why Aquasense?

Real Time Monitoring for Healthier Aquariums

Aquarium health depends on consistent water quality, yet hobbyists often lack real-time, affordable tools to detect dangerous fluctuations.

Limitation of Existing Solutions

Most commercial monitoring systems are costly and tailored for industrial use, making them inaccessible for home aquarium owners.

Affordable

Aquasense fills the gap with a low-cost, Wi-Fi-enabled solution for continuous tracking of key parameters like pH and temperature.

(existing example solution. expensive, no smart-integration)

Proposed Solution (Key Features)

AquaSense: Real-Time, Affordable Aquarium Monitoring

- Low-cost, plug-and-play system built on the ESP32 microcontroller
- Continuously monitors key aquatic parameters (pH, temperature, and lighting conditions)
- Real-time Wi-Fi/Bluetooth connectivity, live data logging, and remote access via user-friendly dashboard
- Automated mobile alerts when conditions become unsafe, along with intelligent tips that uniquely adapt to the user's own aquarium, enabling quick action to protect aquatic life.
- Designed for ease of use and wide accessibility, AquaSense empowers aquarium owners with proactive, reliable water quality oversight.

High Level Requirements

(1) Continuous Monitoring of Critical Water Quality Params

AquaSense will track pH, temperature, and light levels every 5–10 seconds with high accuracy, ensuring timely detection of harmful fluctuations and storing over 1,000 data points per parameter for 30-day analysis.

(2) Wireless Data Transmission

Sensor readings will be wirelessly transmitted via Wi-Fi/Bluetooth to a cloud dashboard within 2 seconds, with alerts (via push/email) triggered when user-defined thresholds are breached.

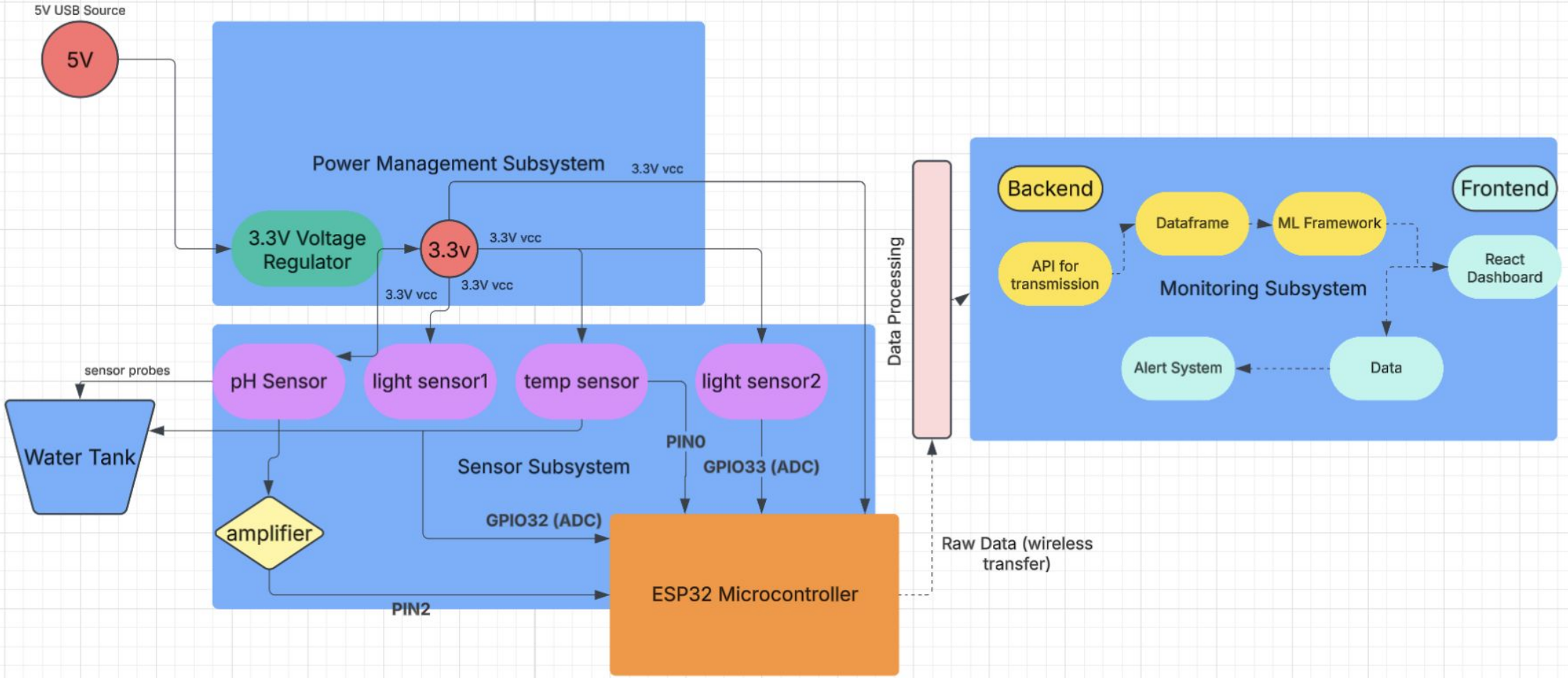
(3) Water Quality Trend Analysis and Machine Learning Based Anomaly Detection

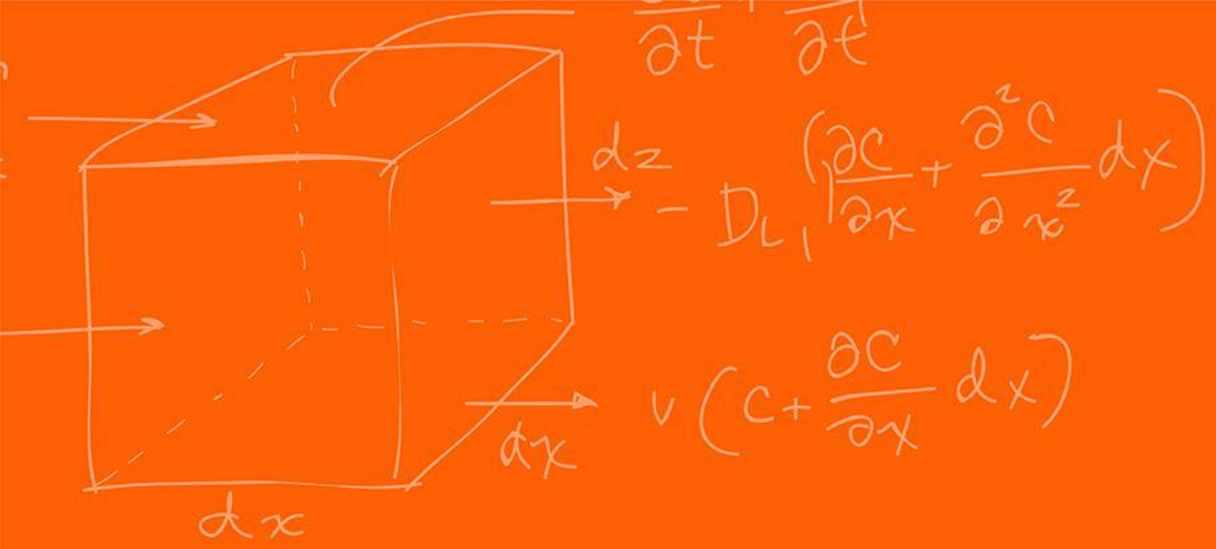
AquaSense will apply time series ML models updated daily to detect deviations from baseline trends, aiming for 90% accuracy in predicting equipment or chemical failures before they become critical.

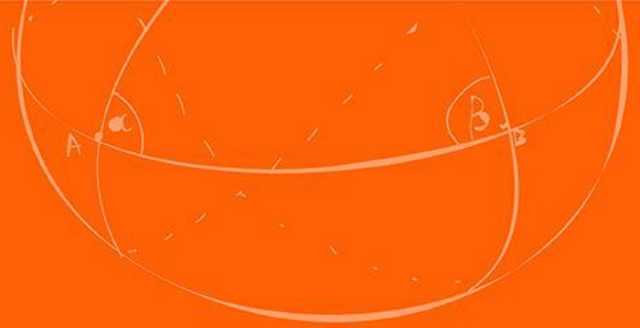
(4) Affordable and Simple Set Up

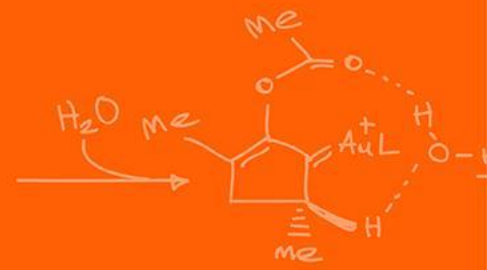
Designed for ease and accessibility, AquaSense features a plug-and-play setup with minimal calibration, offering powerful water monitoring at a fraction of the cost of industrial systems.

Block Diagram



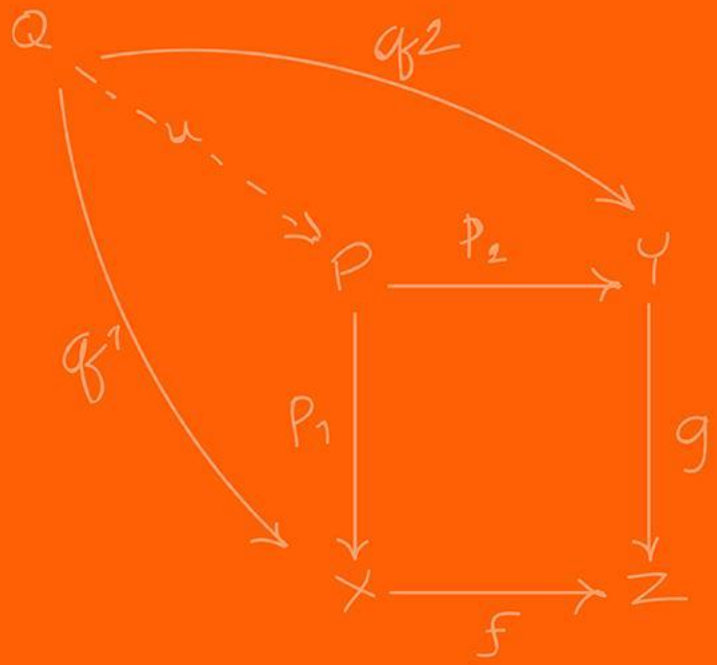






Hardware

Hardware Design + Process



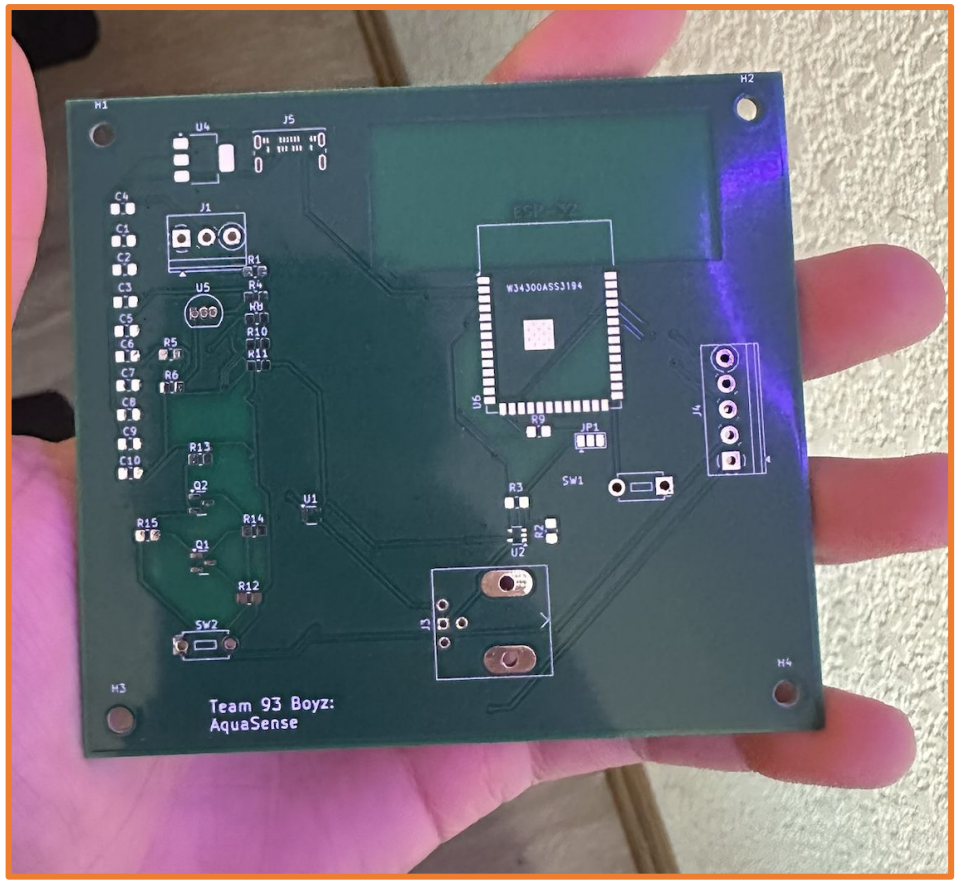
ESP32 (Microcontroller)

Connection Points

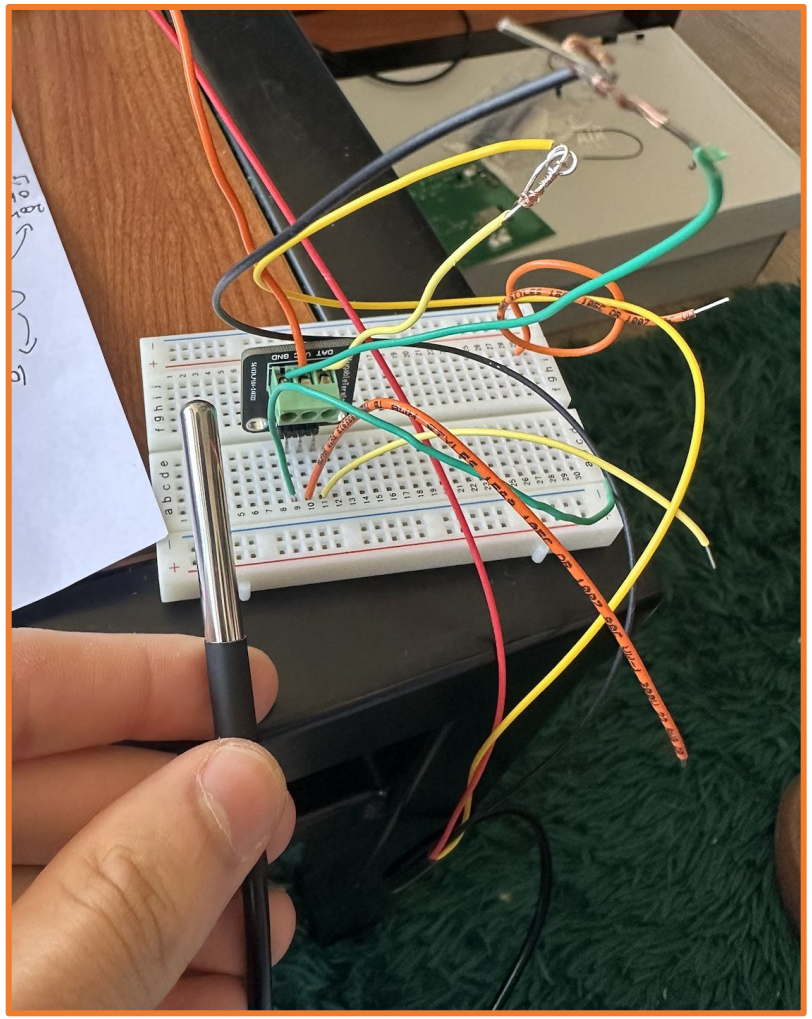
- The ESP32 serves as the core of AquaSense, interfacing with all critical hardware through analog and digital I/O pins, while also managing power input and USB communication.

Core of System Pipeline

- Sensor readings are collected via GPIO interfaces and transmitted in real time to our software backend framework via Wi-Fi/Bluetooth. The ESP32 processes, timestamps, and validates this data before pushing it to the cloud dashboard or local system for visualization and alerts.



Temperature Sensor



DS18B20 Digital Temperature Sensor

- Digital, waterproof sensor chosen for its reliability and ease of integration with ESP32
- ±0.5°C accuracy — ideal for detecting smallest thermal shifts that may affect aquatic life.
- Uses OneWire digital interface, minimizing wiring complexity while supporting multiple sensors on the same line if needed.

Integration & Real-Time Behavior

Connected directly to the ESP32 via GPIO, this sensor captures readings every 5–10 seconds and transmits values wirelessly; alerts are issued within 2 seconds if temperature moves beyond user-defined safe ranges.

Real-Time Transmission

- Must capture and transmit temperature readings every 5–10 seconds. Verified using stopwatch and operating system/epoch-based timestamps collected in the data.

Reading Accuracy

- Sensor temperature measurements were verified against three (3) different thermometers as ground-truth
- Within +/- 0.5 degrees Celsius accuracy (well within our needs of sensitivity for our use case)

pH Sensor


```
    returns:
    - calculated pH value
    """
    R = 8.3145      # Universal gas constant, J/(mol·K)
    F = 96485.0     # Faraday constant, C/mol
    K = temp + 273.15  # C -> K
    slope = (R * K) / F * 2.303  # in Volts per pH unit (~0.05916 at 25°C)

    ph_value = 7 - ((voltage - e0) / slope)
    return ph_value
```

SEN0161 pH Sensor Module

- Analog pH sensor selected for its high sensitivity
- Delivers ±0.2 pH unit accuracy essential for detecting sudden pH shifts that could impact aquatic life.

Calibration and Data Integration

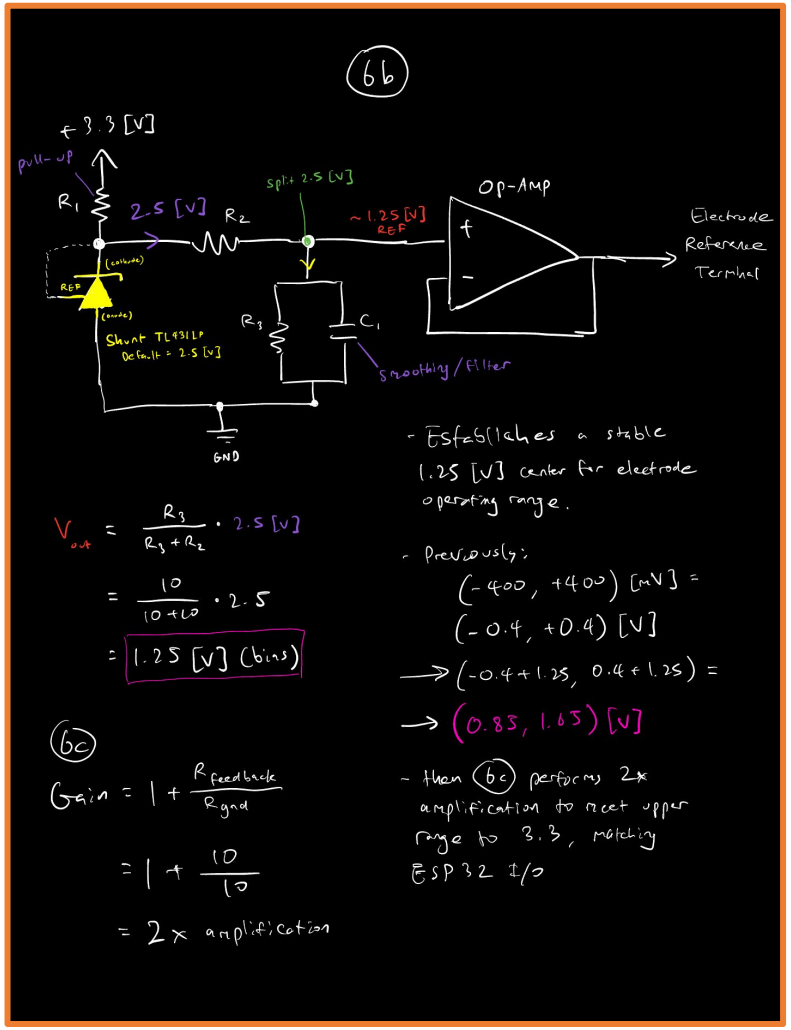
- Sensor delivers analog output, preprocessed via probe conditioning (amp, stabilize, voltage ref)
- System supports two-point calibration for accuracy and issues alerts within 2 seconds of unsafe pH levels.

Real-Time Transmission

- Must capture and transmit accurate pH readings every 5–10 seconds.

Electrode Conditioning

- Due to the nature of pH detection, the raw probe is highly sensitive
- Output signal is very weak
- The system must adequately transform probe readings into a stable and strong enough signal for ESP32
- Hardware signal smoothing and amplification
- Additional software manual calibration



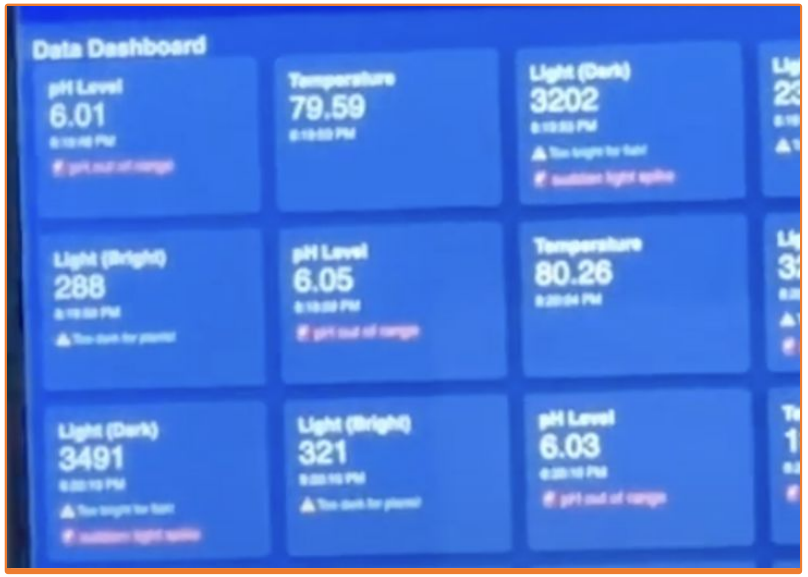
(calculations and design of bias generation for pH reference)

Real-Time Transmission

- Must capture and transmit accurate pH readings every 5–10 seconds.

Electrode Accuracy

- Three known reference samples were used to ensure accuracy of the pH sensor.
  (1) Tap water – 7.5 [pH]
  (2) Cow milk – 6.7 [pH]
  (3) White vinegar – 2.5 [pH]



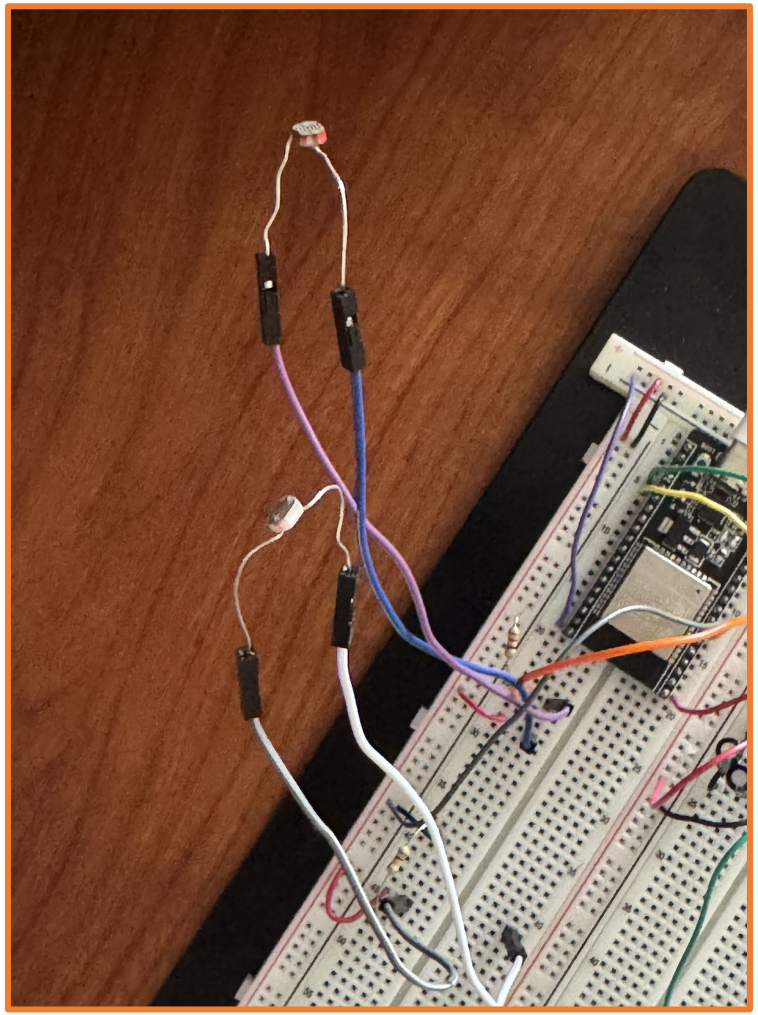
Dual Light Sensor Modes – Fine-Tuned Monitoring

Two photoresistors: one calibrated for low-light, one for bright-light conditions.

- Low-Light Mode: optimized for monitoring nocturnal fish or shaded tanks.
- High-Light Mode: ideal for tracking plant-friendly lighting conditions.

User-selectable modes via dashboard toggle.

Enables accurate light monitoring tailored to fish or plant-focused setups.

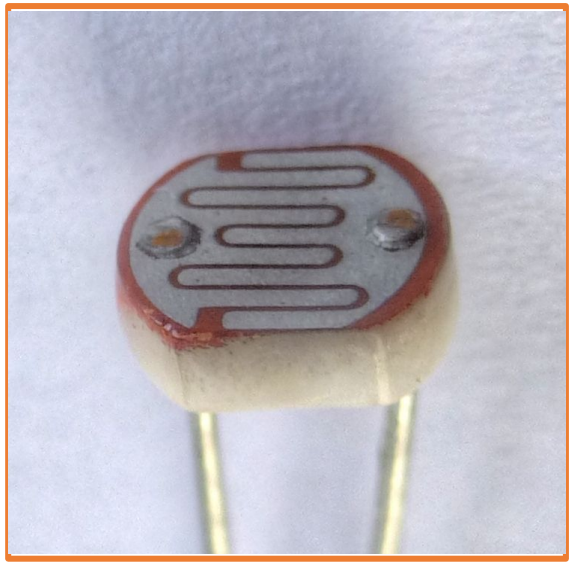


Real-Time Transmission

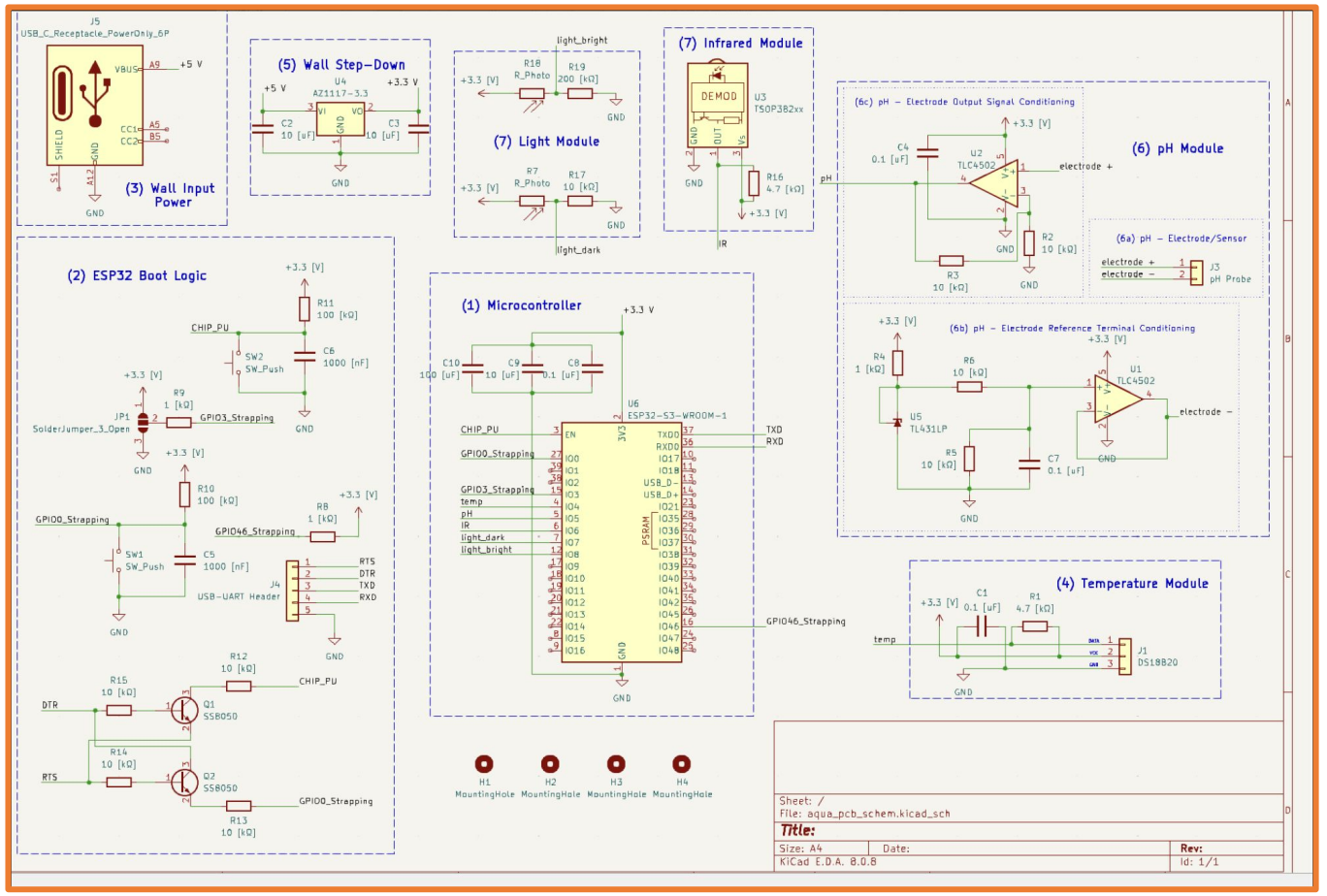
- Must capture and transmit accurate Light readings every 5–10 seconds.

Sensor Accuracy

- ESP32's built in ADC converts the raw voltage changes into a full range of 0 to 4096 digitized reading range.
- Light sensor tested in 3 controlled lighting conditions. Bright (~300 lux), iPhone 15 Pro flashlight (~3000 lux) and fully covered (~10 lux)

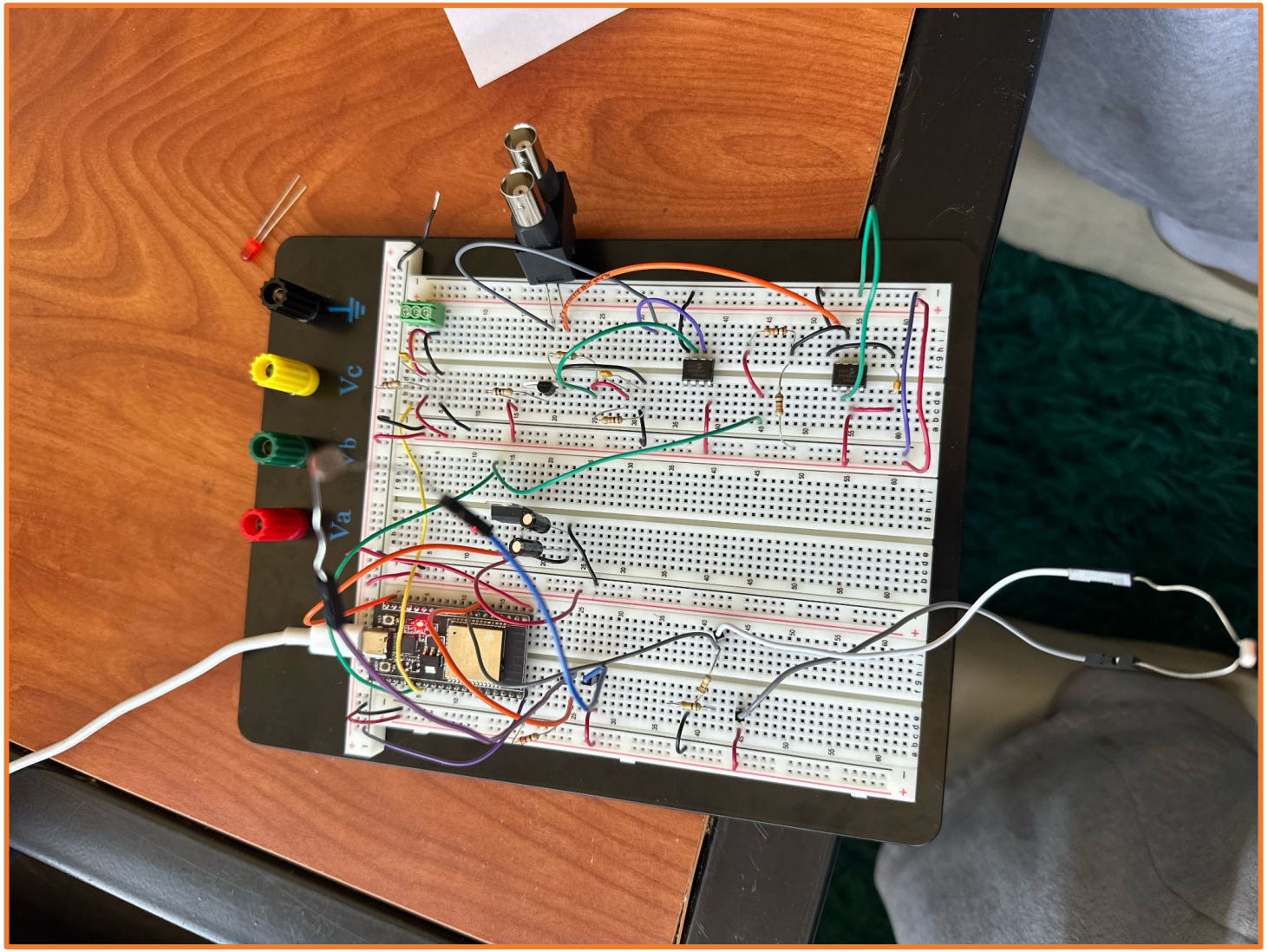


Schematic



Schematic of Aquasense Hardware

Breadboard Setup

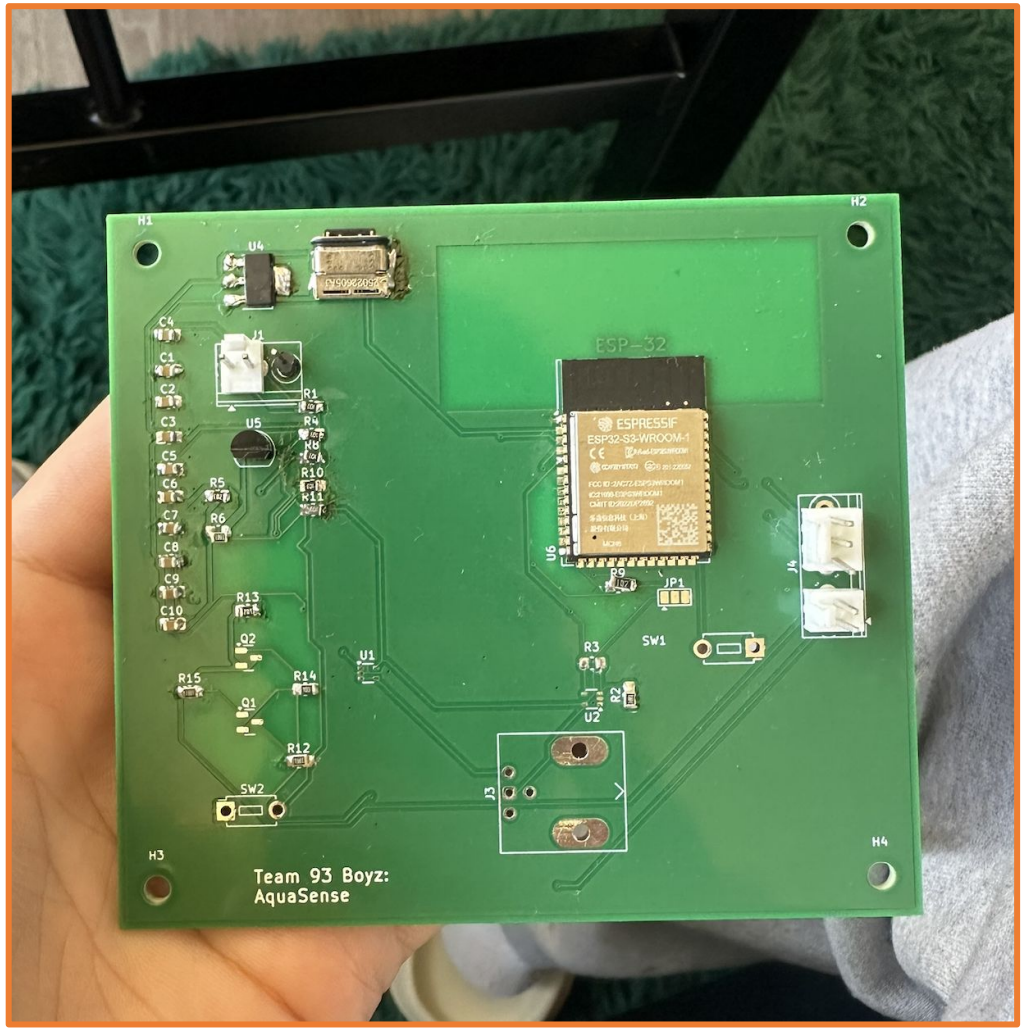


Custom 2-Layer PCB for Modular Sensor Integration

- Integrates all AquaSense subsystems into compact 2-layer board, supporting analog and digital sensors, regulated power input, and seamless ESP32 microcontroller operation. Each system parameter (pH, temperature, light) connects via dedicated GPIOs, allowing for scalability and plug-and-play prototyping.

A look into our process:

- KiCAD + PCBWay → Final System Circuit
- DigiKey, ECEB → Solderable + breadboard parts
- Soldered PCB and Breadboard → Final in-hand product



Description	Manufacturer	Item #	Quantity	Price	Link
ESP32-WROOM-32	Espressif Systems	ESP32-WROOM-32	1	$3.00	link
SEN0161 pH Sensor	Digikey	SEN0161	1	$29.50	link
DS18B20 Digital Temperature Sensor	Maxim Integrated	DS18B20	1	$6.00	link
DFRobot Gravity Dissolved Oxygen Sensor	Atlas Scientific	SEN0237	1	$11.80	link
Analog Turbidity Sensor SEN0189	DFRobot	SEN0189	1	$9.90	link
Op-Amp: UA741CP	Digikey	UA741CP	1	$0.50	link
LM4041 Precision Voltage Reference	Digikey	LM4041	2	$0.50	link
TLA431 Adjustable Shunt Regulator	Digikey	TLA431	2	$0.50	link
VLD1117V33 Voltage Regulator 3.3V	Amazon	VLD1117V33	1	$8.99	link
Fish Tank	Aqueon	100528594	1	$50.90	link
Fish Food (Tropical Flakes)	Tetra	16150	1	$5.50	link
Resistors	Vishay / Yageo	150-piece kit	1	$10.50	link
5mm LED Diodes Kit	Chanzon	B07W8DGBVF	1	$8.50	link
Buzzer 5V	Adafruit	1536	1	$3.99	link

Parts List

Functional Blocks and Power Architecture

The schematic includes key modules:

- (1) ESP32-S3-WROOM-1 for processing and wireless communication
- (3) Wall input power and (5) voltage step-down to 3.3V
- (4) Temperature module using DS18B20
- (6) pH signal conditioning with TLC4502 op-amps
- (7) Light and IR detection modules

All sensors are routed with decoupling capacitors and pull-up resistors where required, ensuring stable signal acquisition.

Middleware (Arduino Sensor Subsystem)

Arduino Hardware Interface

Written in C++ and integrated with ESP32 APIs, this interface captures readings from all connected sensors and transmits the data via Wi-Fi to the backend in 5–10 second intervals.

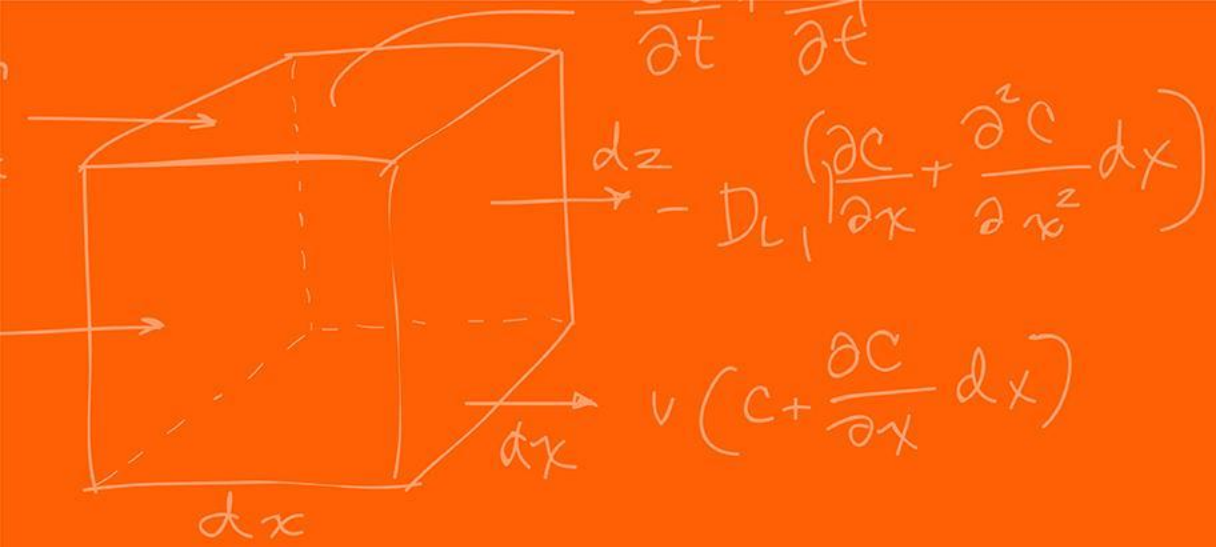
```
void loop() {
  if (WiFi.status() == WL_CONNECTED) {
    // --- Temperature POST ---
    HTTPClient http;
    http.begin(tempServerUrl);  // Temp still goes to /data/temp
    http.addHeader("Content-Type", "application/json");

    temp_sensor.requestTemperatures();
    float data_temp = temp_sensor.getTempFByIndex(0);

    String jsonPOST_temp = "{\"sensor_type\": \"temp\", \"value\": " + String(data_temp) + "}";
    int httpResponseCode_temp = http.POST(jsonPOST_temp);

    http.end(); // Close temp HTTP session

    // --- Light Sensors POST ---

    float data_light_dark = analogRead(light_sensor_dark);
    float data_light_bright = analogRead(light_sensor_light);

    HTTPClient http_light_dark;
    http_light_dark.begin(lightServerUrl);
    http_light_dark.addHeader("Content-Type", "application/json");

    String jsonPOST_light_dark = "{\"sensor_type\": \"light_dark\", \"value\": " + String(data_light_dark) + "}";
    int httpResponseCode_light_dark = http_light_dark.POST(jsonPOST_light_dark);

    http_light_dark.end(); // close after POST

    HTTPClient http_light_bright;
    http_light_bright.begin(lightServerUrl);
    http_light_bright.addHeader("Content-Type", "application/json");

    String jsonPOST_light_bright = "{\"sensor_type\": \"light_bright\", \"value\": " + String(data_light_bright)
```

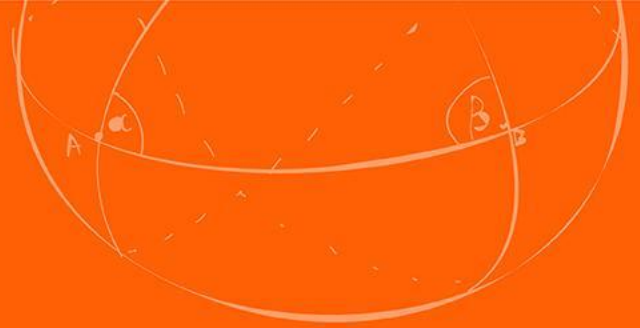
Data Preprocessing

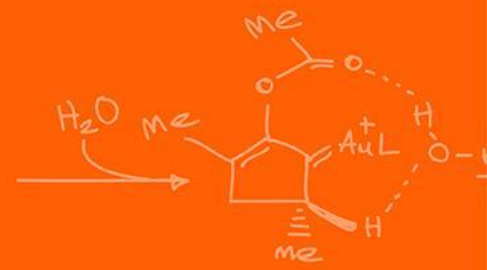
- Take raw sensor voltage outputs and prepare for sending to backend via HTTP routes → JSON response

RV: Real-Time Transmission

- Must capture and transmit pH, temperature, and light readings every 5–10 seconds.
- Easily verifiable with stopwatch

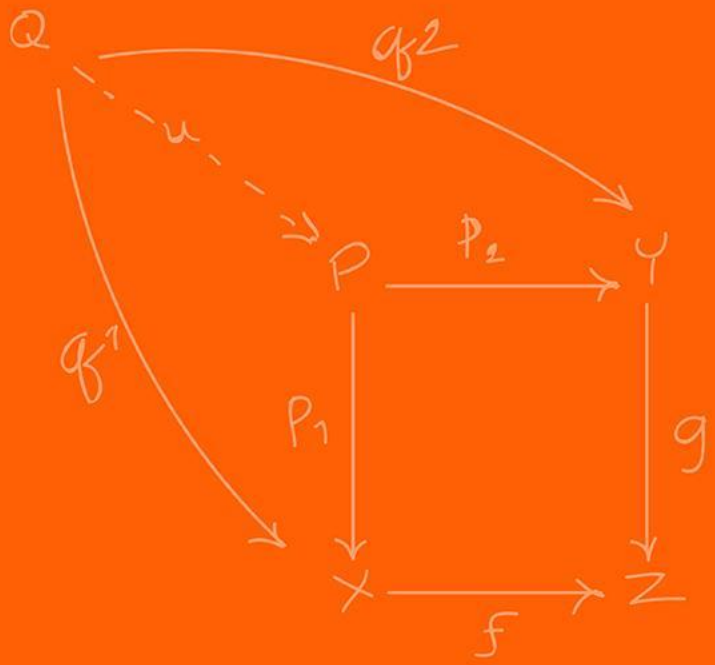






Software

Hardware Schematic + Design



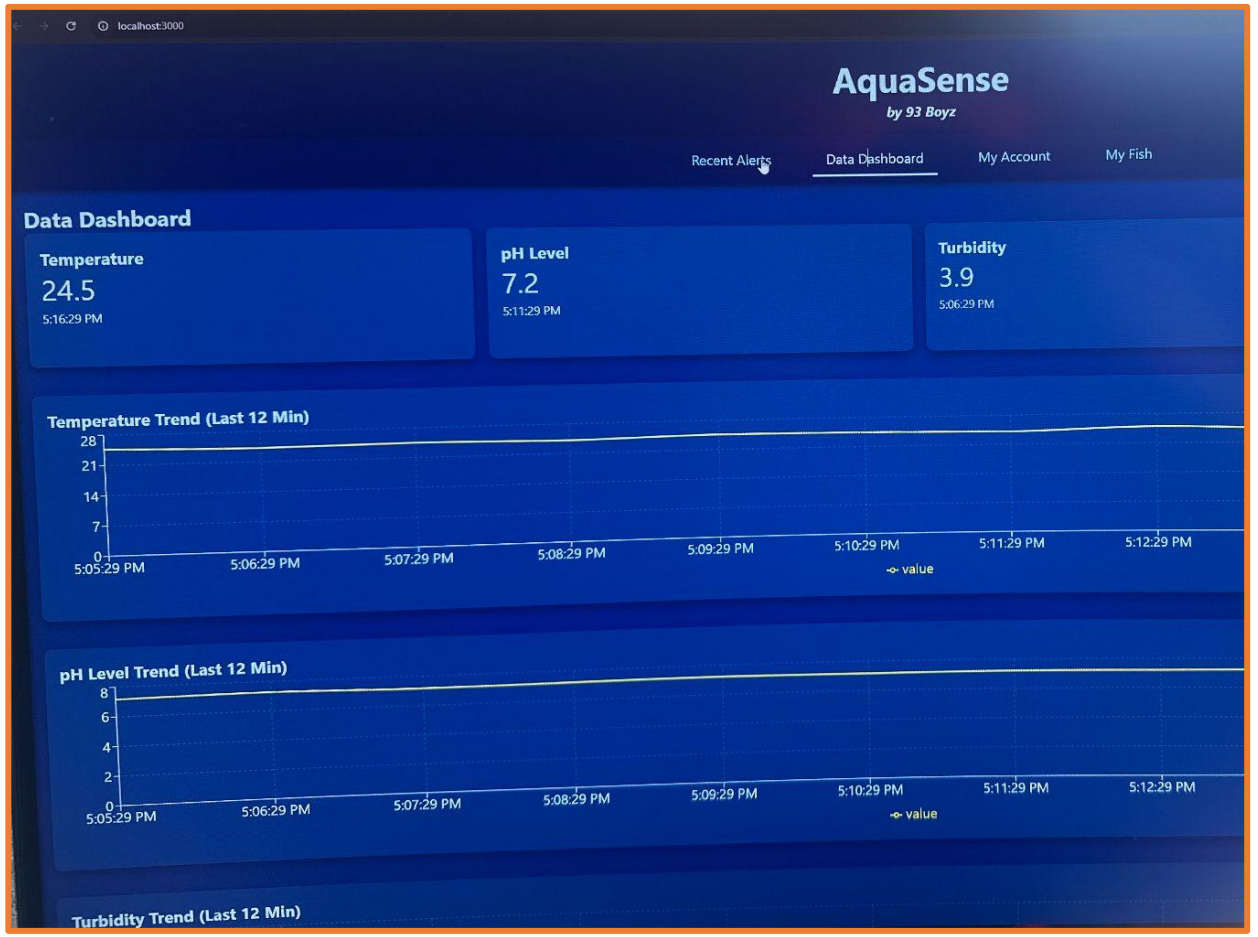
Frontend

Web Dashboard for Real-Time Monitoring

Built using Vite, React, and Node.js, the frontend serves as the primary interface for users to view real-time sensor readings, historical trends, and water quality alerts.

Interactive & Modular UI

Users can configure thresholds, visualize anomalies with graphs, and view summaries for each monitored parameter. The dashboard also adapts to multiple AquaSense nodes and supports responsive design for desktops and tablets.



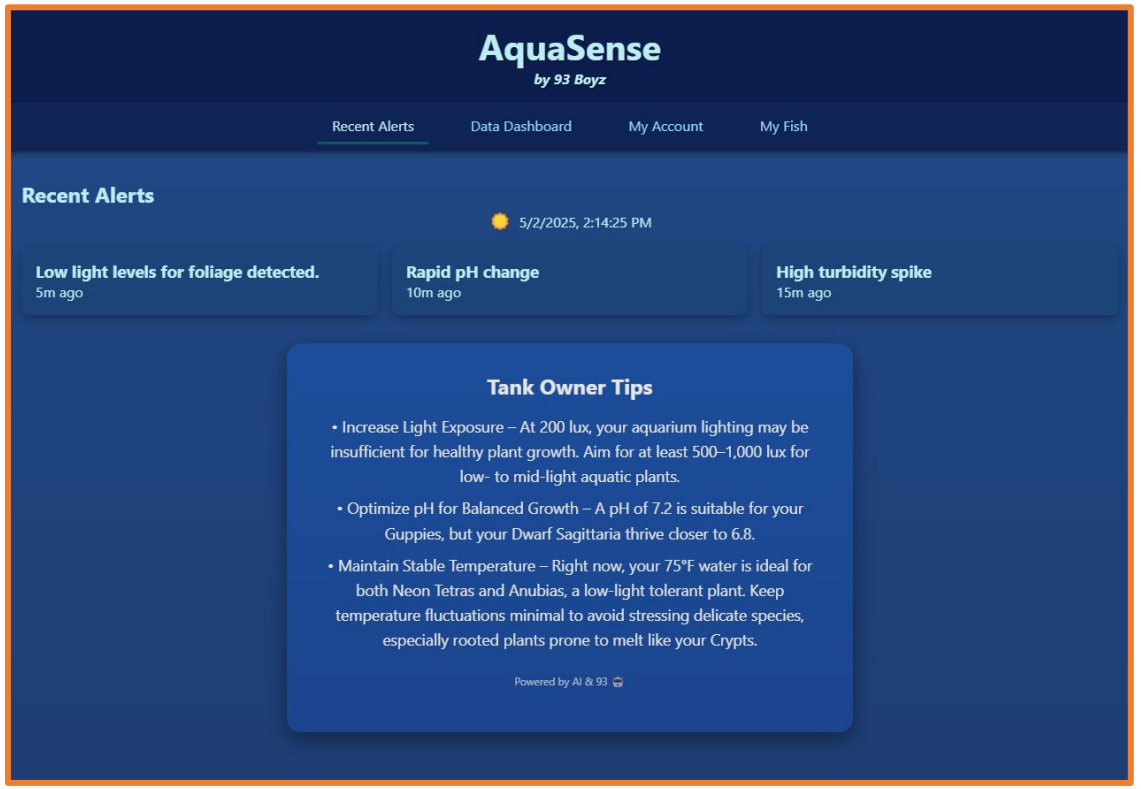
RV: System Monitor

(1) Real-Time User Interface

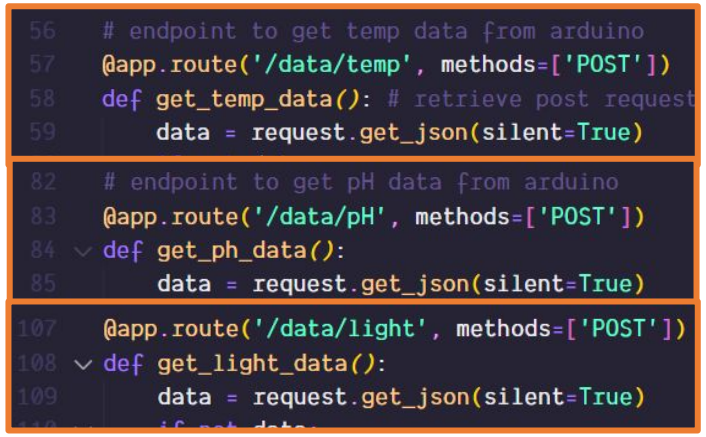
- Must display real-time sensor values to the user at 10-second intervals or less, with an end-to-end latency of under 2 seconds from measurement to dashboard update.

(2) Extract Meaning from Dense Data

- ML algorithms must achieve high accuracy in identifying critical events such as heater failures, and give context-aware responses that the user can use to learn how to properly care for their ecosystem.
- (see image to the right for verification)



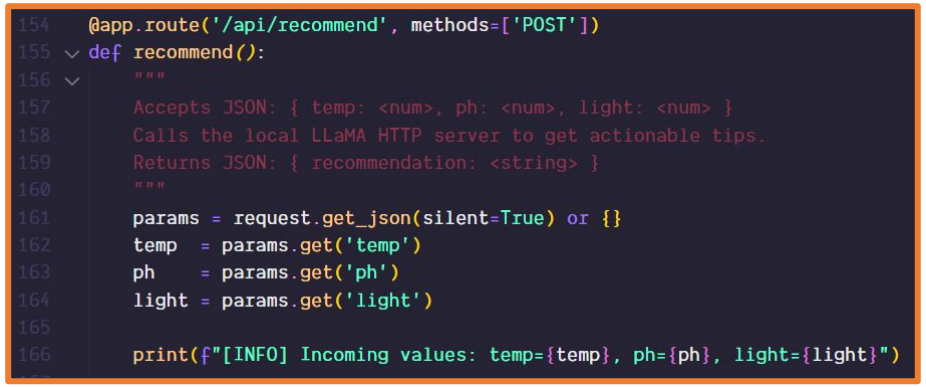
Backend



```python
# endpoint to get temp data from arduino
@app.route('/data/temp', methods=['POST'])
def get_temp_data(): # retrieve post request
    data = request.get_json(silent=True)
```

```python
# endpoint to get pH data from arduino
@app.route('/data/pH', methods=['POST'])
def get_ph_data():
    data = request.get_json(silent=True)
```

```python
@app.route('/data/light', methods=['POST'])
def get_light_data():
    data = request.get_json(silent=True)
```



```python
@app.route('/api/recommend', methods=['POST'])
def recommend():
    """
    Accepts JSON: { temp: <num>, ph: <num>, light: <num> }
    Calls the local LLaMA HTTP server to get actionable tips.
    Returns JSON: { recommendation: <string> }
    """
    params = request.get_json(silent=True) or {}
    temp  = params.get('temp')
    ph    = params.get('ph')
    light = params.get('light')

    print(f"[INFO] Incoming values: temp={temp}, ph={ph}, light={light}")
```

Sensor Data Pipeline (Python + Flask)

- Flask-based Python backend that handles all communication between the frontend and hardware
- Receives, logs, and stores into time-series database
- Also triggers alerts based on threshold violations and statistical pattern mining over database.

API Layer for Frontend Access

Exposes endpoints for real-time and historical data access, serving the frontend with processed sensor values, trends, and alert history. All data is updated in ~2-second intervals.

RV: Backend Data Framework

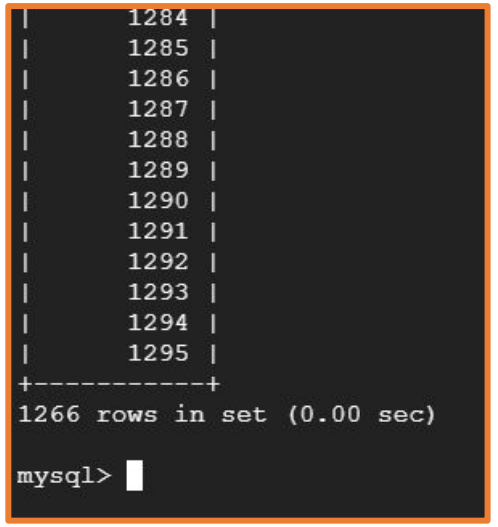
Data Persistence

- Data logging must capture at least 1,000 data points per parameter over a 30-day rolling window to enabling machine learning algorithms to detect anomalies and perform strong inference.

MySQL Database via Google Cloud

- DB hosted on GCP SQL instance to achieve persistence across sessions and allow for continuous model training and long-term data analysis

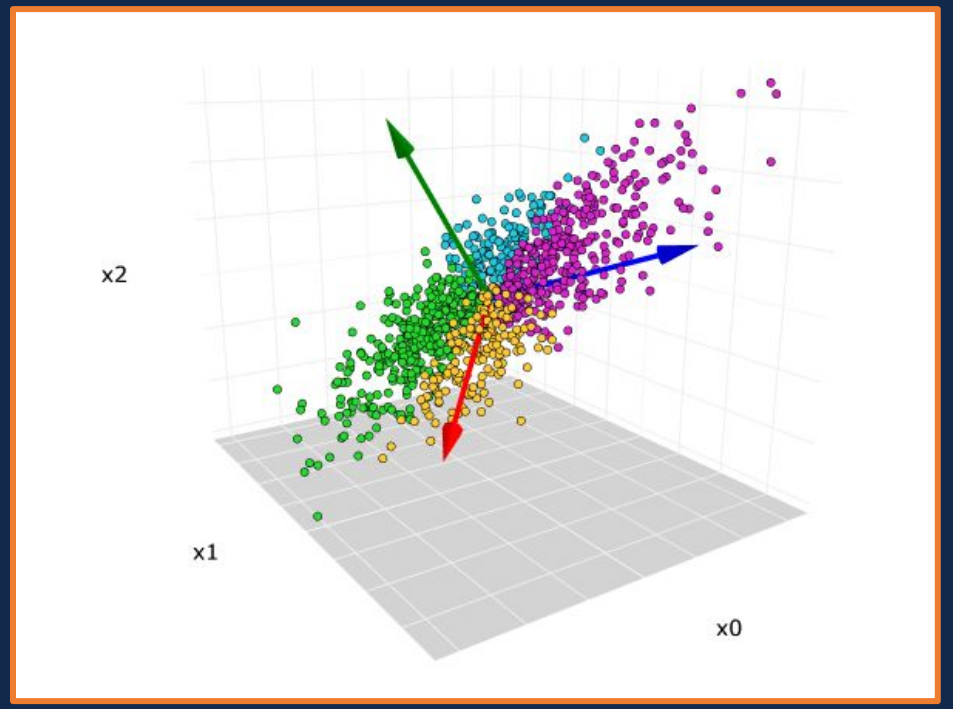
Timestamp	pH	Temperature (°C	ssolved O₂ (mg
2025-04-08 8:00	7.45	26.1	6.7
2025-04-08 8:00	7.42	26	6.5
2025-04-08 8:00	7.4	26	6.6
2025-04-08 8:00	7.38	25.9	6.4
2025-04-08 8:00	7.37	25.9	6.3
...	...	...	...
2025-05-07 18:3	7.51	25.8	6.8
2025-05-07 18:3	7.49	25.9	6.6
2025-05-07 18:3	7.47	26	6.5
2025-05-07 18:3	7.46	26.1	6.7
2025-05-07 18:3	7.44	26.1	6.8



Unsupervised Statistical Machine Learning

PCA and Dimensionality Reduction

Compress time-variant data into easily operable dimension space, maintaining key statistical quantities and highlighting



FPGrowth Association Rule Mining

- Statistically discovers co-occurring data events that lead to failures or anomalies previously found from PCA.
- Converts event logs into frequent itemsets, condense data across thousands of timestamps into easily interpretable ‘pattern’ tokens for the LLM to handle
- Unsupervised nature allows the entire engine to be self-sufficient and run in a feedback loop without manual intervention

LLM Pipeline

```
You are an expert aquarium consultant…
- Water temperature: 75 °F
- pH level: 7.2
- Brightness: 200 lux

Provide three concise tips…
```

```
# Tokenize and move to model
inputs = tokenizer(prompt, return_tensors="pt")
outputs = model.generate(
    **inputs,
    max_new_tokens=100,
    do_sample=True,
    top_p=0.9,
    temperature=0.8,
    pad_token_id=tokenizer.eos_token_id
)

# Decode full output (prompt + generated) and strip the prompt
full = tokenizer.decode(outputs[0], skip_special_tokens=True)
continuation = full[len(prompt):].strip()

return jsonify({"text": continuation})
```

```
# Load a compact causal model for on-device inference
MODEL_NAME = "distilgpt2"
tokenizer = AutoTokenizer.from_pretrained(MODEL_NAME)
model     = AutoModelForCausalLM.from_pretrained(MODEL_NAME)
```

LLM Server

This server hosts our custom recommendation engine. When anomalies are detected (e.g., sudden pH drop), it provides context-aware suggestions like "*Perform a partial water change.*"

LLM Integration

The system uses basic rule-based logic augmented with templated LLM prompts to translate numeric deviations into actionable advice personalized for fish type and tank size.

Challenges

Sensor Selection and Budget

- Balancing cost and accuracy in sourcing sensors that can deliver ample parameters for ML inference and user suggestions
- Had to scrap original Dissolved Oxygen idea, probe is way out of budget

Conditioning Circuit/Schematic Debugging

- Exhaustive tuning analog front-ends to ideal values and arrangements (op-amps, shunts, transistors)
- Issues with PCB microcontroller logic not functioning the same as schematic/breadboard

ML Integration

- Experimented with various models (XGBoost, Random Forest, K-Means, NN's, Apriori) before settling on PCA/FPGrowth
- Had to do major rework on backend data structures to properly transform probe inputs in order to pass into ML models

Looking to the Future

Potential New Parameters

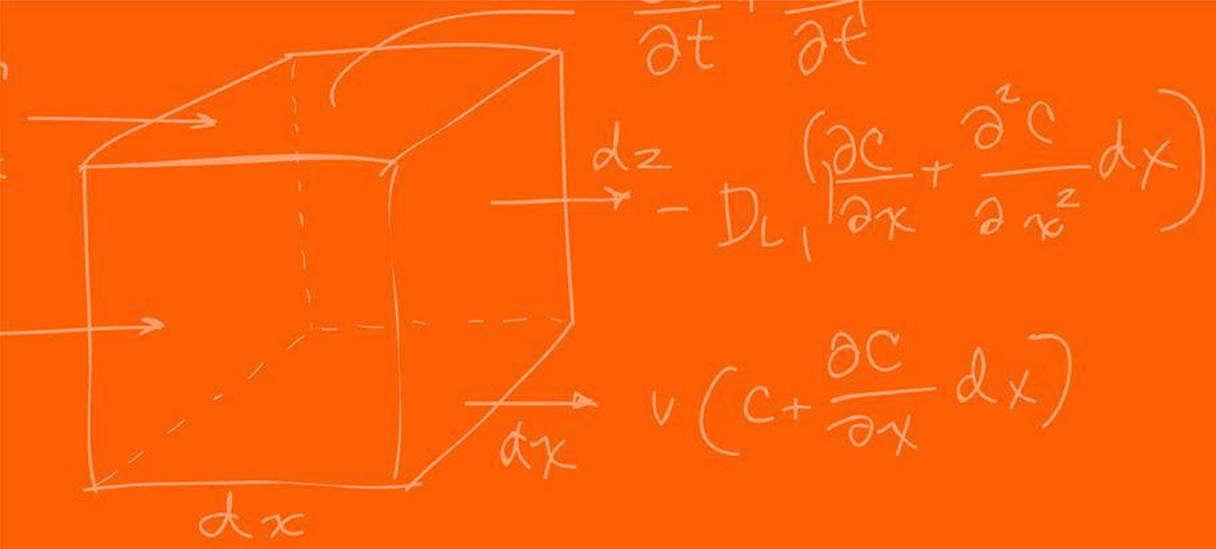
- Ammonia MQ137 Sensor (~$14, replicable breakout)
- Nitrate sensor (may be out of reasonable budget)

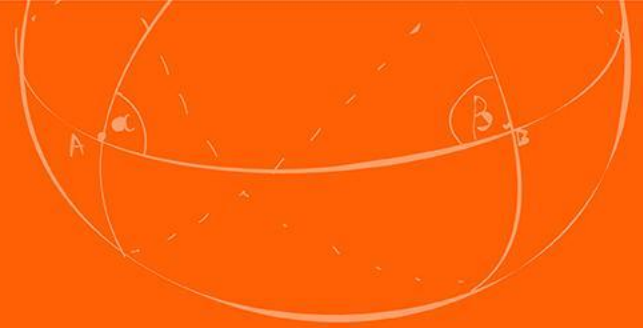
Mobile App Buff

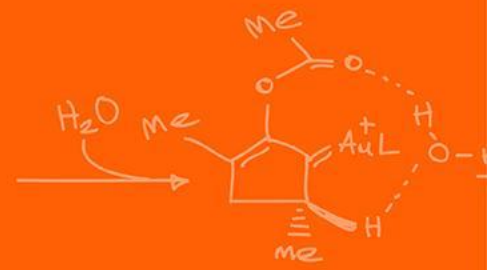
Make an optimized dashboard for mobile devices, taking advantage of their ease-of-access and usability for hobbyists

Addressing PCB Issues

Troubleshooting existing issues with our PCB design, like loading program images onto the microcontroller (likely incorrect BOOT/EN/IO0 state timing or wrong serial conns)







The Grainger College of Engineering

UNIVERSITY OF ILLINOIS URBANA-CHAMPAIGN